ELSEVIER

Theoretical Computer Science 295 (2003) 41–64

Theoretical Computer Science

www.elsevier.com/locate/tcs

Reachability problems for sequential dynamical systems with threshold functions☆

Chris Barrett[a,1], Harry B. Hunt III[b,2], Madhav V. Marathe[a,1], S.S. Ravi[b,2,*], Daniel J. Rosenkrantz[b,2], Richard E. Stearns[b,2]

[a] *Los Alamos National Laboratory, MS M997, P.O. Box 1663, Los Alamos, NM 87545, USA*

[b] *Department of Computer Science, University at Albany—SUNY, 1400 Washington Ave., Albany, NY 12222, USA*

Received 27 September 2001; received in revised form 6 March 2002; accepted 14 March 2002

Abstract

A sequential dynamical system (SDS) over a domain $\mathbb{D}$ is a triple $(G, \mathscr{F}, \pi)$, where (i) $G(V,E)$ is an undirected graph with n nodes with each node having a state value from $\mathbb{D}$, (ii) $\mathscr{F} = \{f_1, f_2, \ldots, f_n\}$ is a set of local transition functions with f_i denoting the local transition function associated with node v_i and (iii) π is a permutation of (i.e., a total order on) the nodes in V. A single SDS transition is obtained by updating the states of the nodes in V by evaluating the function associated with each of them in the order given by π.

We consider reachability problems for SDSs with restricted local transition functions. Our main intractability results show that the reachability problems for SDSs are *PSPACE*-complete when either of the following restrictions hold: (i) $\mathscr{F}$ consists of both simple-threshold-functions and simple-inverted-threshold functions, or (ii) $\mathscr{F}$ consists only of threshold-functions that use weights in an asymmetric manner. Moreover, the results hold even for SDSs whose underlying graphs have bounded node degree and bounded pathwidth. Our lower bound results also extend to reachability problems for Hopfield networks and communicating finite state machines.

On the positive side, we show that when $\mathscr{F}$ consists only of threshold functions that use weights in a symmetric manner, reachability problems can be solved efficiently provided all the

☆ A preliminary version of some of the results in this paper were reported in [3].

* Corresponding author.

E-mail addresses: barrett@lanl.gov (C. Barrett), hunt@cs.albany.edu (H.B. Hunt III), marathe@lanl.gov (M.V. Marathe), ravi@cs.albany.edu (S.S. Ravi), djr@cs.albany.edu (D.J. Rosenkrantz), res@cs.albany.edu (R.E. Stearns).

[1] Research supported by the US Department of Energy under Contract W-7405-ENG-36.

[2] A part of this work was carried out when Harry Hunt and S.S. Ravi were visiting the Basic and Applied Simulation Sciences Group (D-2) of the Los Alamos National Laboratory. This research was supported by NSF Grants CCR-01-05336 and CCR-97-34936 and a grant from the Los Alamos National Laboratory.

PII: S0304-3975(02)00395-X

weights are strictly positive and the ratio of the largest to the smallest weight is bounded by a polynomial function of the number of nodes.

MSC: 68Q10; 68Q17; 68Q80

Keywords: Computational complexity; Theory of simulation; Dynamical systems; Hopfield networks; Communicating finite state machines; Cellular automata

1. Introduction

1.1. Definition of a sequential dynamical system

We study the computational complexity of reachability problems for *sequential dynamical systems* (SDSs), a new class of discrete finite dynamical systems introduced in [5,26]. Formally, a sequential dynamical system $\mathscr{S}$ over a given domain $\mathbb{D}$ of state values is a triple $(G,\mathscr{F},\pi)$, whose components are as follows:

(1) $G(V,E)$ is a finite undirected graph without multi-edges or self loops. G is referred to as the *underlying graph* of $\mathscr{S}$. We use n to denote $|V|$ and m to denote $|E|$. The nodes of G are numbered using the integers $1,2,\ldots,n$.
(2) For each node i of G, $\mathscr{F}$ specifies a *local transition function*, denoted by f_i. This function maps $\mathbb{D}^{\delta_i+1}$ into $\mathbb{D}$, where δ_i is the degree of node i. Letting $N(i)$ denote the set consisting of node i itself and its neighbors, each parameter of f_i corresponds to a member of $N(i)$.
(3) Finally, π is a permutation of $\{1,2,\ldots,n\}$ specifying the order in which nodes update their states using their local transition functions. Alternatively, π can be envisioned as a total order on the set of nodes.

A *configuration* $\mathscr{C}$ of $\mathscr{S}$ can be interchangeably regarded as an n-vector $(c_1,c_2,\ldots,c_n)$, where each $c_i\in\mathbb{D}$, $1\leqslant i\leqslant n$, or as a function $\mathscr{C}:V\to\mathbb{D}$. From the first perspective, c_i is the state value of node i in configuration $\mathscr{C}$, and from the second perspective, $\mathscr{C}(i)$ is the state value of node i in configuration $\mathscr{C}$.

Computationally, each step of an SDS (i.e., the transition from one configuration to another), involves n substeps, where the nodes are processed in the *sequential* order specified by permutation π. The "processing" of a node consists of computing the value of the node's local transition function and changing its state to the computed value. The following pseudocode shows the computations involved in one transition:

---

for $i=1$ **to** n **do**

(i) Node $\pi(i)$ evaluates $f_{\pi(i)}$. (This computation uses the *current* values of the state of $\pi(i)$ and those of the neighbors of $\pi(i)$.) Let x denote the value computed.

(ii) Node $\pi(i)$ sets its state $s_{\pi(i)}$ to x.

end-for

1.2. Motivation

Our primary motivation for studying SDSs is to develop an axiomatic theory of simulation systems which can be applied to the design of large scale socio-technical simulations of national infrastructures such as transportation, electrical power and communication. The SDS model has been successfully used in the design of a large-scale transportation simulation system called TRANSIMS[3] at the Los Alamos National Laboratory.

Another motivation for studying SDSs is that they are closely related to some well-known models used in dynamical systems, machine learning and distributed computing. Thus, lower bounds on the computational complexity of deciding some properties of SDSs yield as direct corollaries analogous results for those models. The models include the following:

(a) Classical cellular automata (CA) [37] and *graph automata* [27,23], which are a widely studied class of dynamical systems in physics and complex systems.

(b) Discrete Hopfield networks [19,10], which are a classical model for machine learning, and

(c) Communicating finite state machines [1,14,18,24], which are widely used to model and verify distributed systems.

The main difference between graph automata and SDSs is that in the former, node states are updated in parallel while in latter, they are updated in a specified sequential order. Recently, other authors [20,11,32] have also considered sequential updates. In particular, Huberman and Glance [20] present experimental results to show that certain simulations of n-person games exhibit very different (but probably more realistic) dynamics when the cells are updated sequentially as opposed to when they are updated in parallel.

Decidability issues for dynamical systems in general and for CA in particular have been widely studied in the literature (see for example, the two edited volumes [37,16]). In contrast, computational complexity questions arising in the study of finite CA and related dynamical systems have received comparatively less attention [35,33,15,13]. Here we study the computational complexity of several reachability problems for SDSs. These are fundamental problems in the context of analyzing dynamical systems. Our results indicate that these questions are, in general, computationally intractable. However, we identify several special classes of SDSs for which the questions can be answered efficiently.

2. Terminology and problem definitions

2.1. Additional SDS terminology

Let $\mathscr{S}=(G,\mathscr{F},\pi)$ denote an SDS over a domain $\mathbb{D}$. We let $F_{\mathscr{S}}$ denote the *global transition function* associated with $\mathscr{S}$. For any configuration $\mathscr{C}$, $F_{\mathscr{S}}(\mathscr{C})$ gives the

[3] TRANSIMS is an acronym for the "TRansportation ANalysis and SIMulation System". For details, see http://transims.tsasa.lanl.gov.

configuration reached by $\mathscr{S}$ in one step starting from $\mathscr{C}$. This function can be viewed either as a function that maps $\mathbb{D}^n$ into $\mathbb{D}^n$ or as a function that maps $\mathbb{D}^V$ into $\mathbb{D}^V$. $F_{\mathscr{S}}$ represents the transitions between configurations, and can therefore be considered as defining the dynamic behavior of SDS $\mathscr{S}$. Recall that a configuration $\mathscr{C}$ can be viewed as a function that maps V into $\mathbb{D}$. As a slight extension of this view, we use $\mathscr{C}(W)$ to denote the states of the nodes in $W \subseteq V$.

Let $\mathscr{I}$ denote the designated configuration of $\mathscr{S}$ at time 0. Starting with $\mathscr{I}$, the configuration of $\mathscr{S}$ after t steps (for $t \geqslant 0$) is denoted by $\xi(\mathscr{S}, \mathscr{I}, t)$. Note that $\xi(\mathscr{S}, \mathscr{I}, 0) = \mathscr{I}$ and $\xi(\mathscr{S}, \mathscr{I}, t+1) = F_{\mathscr{S}}(\xi(\mathscr{S}, \mathscr{I}, t))$. Consequently, for all $t \geqslant 0$, $\xi(\mathscr{S}, \mathscr{I}, t) = F^t_{\mathscr{S}}(\mathscr{I})$.

A *fixed point* of an SDS $\mathscr{S}$ is a configuration $\mathscr{C}$ such that $F_{\mathscr{S}}(\mathscr{C}) = \mathscr{C}$. An SDS $\mathscr{S}$ is said to *cycle* through a (finite) sequence of configurations $\langle \mathscr{C}_1, \mathscr{C}_2, \ldots, \mathscr{C}_r \rangle$ if $F_{\mathscr{S}}(\mathscr{C}_1) = \mathscr{C}_2$, $F_{\mathscr{S}}(\mathscr{C}_2) = \mathscr{C}_3, \ldots, F_{\mathscr{S}}(\mathscr{C}_{r-1}) = \mathscr{C}_r$ and $F_{\mathscr{S}}(\mathscr{C}_r) = \mathscr{C}_1$. A fixed point is a cycle involving only one configuration.

The *phase space* $\mathscr{P}_{\mathscr{S}}$ of an SDS $\mathscr{S}$ is a directed graph defined as follows: There is a node in $\mathscr{P}_{\mathscr{S}}$ for each configuration of $\mathscr{S}$. There is a directed edge from a node representing configuration $\mathscr{C}$ to that representing configuration $\mathscr{C}'$ if $F_{\mathscr{S}}(\mathscr{C}) = \mathscr{C}'$. In such a case, we also say that configuration $\mathscr{C}$ is a *predecessor* of configuration $\mathscr{C}'$. Since SDSs are deterministic, each node in its phase space has an outdegree of 1. In general, the phase space $\mathscr{P}_{\mathscr{S}}$ may have an infinite number of nodes. When the domain $\mathbb{D}$ of state values is finite, the number of nodes in the phase space is $|\mathbb{D}|^n$.

2.2. Variations of the basic SDS model

The above definition of an SDS imposes no restrictions on either the domain $\mathbb{D}$ of state values or the local transition functions, except that the range of each local transition function must be a subset of $\mathbb{D}$. SDSs that model simulation systems can be obtained by appropriately restricting $\mathbb{D}$ and/or the local transition functions. We use the notation "(x, y)-SDS" to denote an SDS where 'x' specifies the restriction on the domain and 'y' specifies the restriction on the local transition functions. Some restrictions studied in this paper are discussed below. Whenever possible, we prove our hardness results for the most restricted SDS model, thereby obtaining stronger lower bound results.

We assume that the Boolean domain consists of the two integers 0 and 1. The focus of this paper is on SDSs over the Boolean domain with special classes of Boolean local transition functions. We provide below the definitions (from [21]) of these special classes and also introduce the notation for the corresponding restricted class of SDSs.

Definition 2.1. A *symmetric* Boolean function is one whose value does not depend on the order in which the inputs are specified; that is, the function value depends only on how many of its inputs are 1.

Definition 2.2. Given two Boolean vectors $X = \langle x_1, x_2, \ldots, x_q \rangle$ and $Y = \langle y_1, y_2, \ldots, y_q \rangle$, define the relation "$\leqslant$" as follows: $X \leqslant Y$ if $x_i \leqslant y_i$, $1 \leqslant i \leqslant q$. A q-input Boolean function f is *monotone* if $X \leqslant Y$ implies that $f(X) \leqslant f(Y)$.

Definition 2.3. A *k-simple-threshold function* takes on the value 1 if at least k of the Boolean inputs have value 1; otherwise, the value of the function is 0. A *k-simple-inverted-threshold function* is the negation of the k-simple-threshold function.

Threshold functions, which are a generalization of simple-threshold functions, are defined as follows.

Definition 2.4. A q-input *threshold function* has q Boolean inputs $x_1, x_2, \ldots, x_q$ with respective weights $w_1, w_2, \ldots, w_q$, a Boolean output y and a threshold α. The value of y is 1 iff $\sum_{i=1}^{q} w_i x_i \geqslant \alpha$.

Barrett et al. [5], Mortveit and Reidys [26], Reidys [31] and Laubenbacher and Pareigis [22] investigated mathematical properties of SDSs over the Boolean domain with symmetric local transition functions. Indeed, in [5,26], where the SDS model was introduced, it was assumed that the state values are Boolean and that the local transition functions are symmetric. In our notation, an SDS over the Boolean domain with symmetric local transition functions is referred to as a (BOOL, SYM)-SDS. The use of symmetric functions is one way of capturing "mean field effects" in statistical physics and other large-scale systems. A similar assumption has been made in [7]. When the local transition functions are monotone (but not necessarily symmetric), we denote the corresponding class of SDSs by (BOOL, MON)-SDS.

We use the notation (BOOL, ST)-SDS for the class of SDSs where each local transition function is a simple-threshold function. When the set of local transition functions of an SDS is allowed to consist of both simple-threshold and simple-inverted-threshold functions, the resulting class of SDSs is denoted by (BOOL, SIT)-SDS.

The class of SDSs over the Boolean domain where each local transition function is a threshold function is denoted by (BOOL, AWT)-SDS. In such SDSs, the weights used in the local transition functions at the two nodes of an edge may not be equal. A useful subclass of (BOOL, AWT)-SDSs are those in which the weights used in local transition functions are *symmetric*; that is, for each edge, the weights used by the local transition functions at the two nodes of the edge are equal. We denote this subclass of SDSs by (BOOL, SWT)-SDSs. As will be seen, permitting threshold functions that use weights in an asymmetric manner changes the complexity of reachability problems significantly.

It is also of interest to consider dynamical system models obtained by modifying some components of an SDS. One such model is a *synchronous dynamical system* (SyDS), which is an SDS *without* the node permutation. In a SyDS, during each time step, all the nodes *synchronously* evaluate their local transition functions and update their state values. Thus, SyDSs are similar to classical CA with the difference that the connectivity between cells is specified by an arbitrary graph. The restrictions on the domain of state values and local transition functions discussed for SDSs are also applicable to SyDSs.

Table 1 summarizes the notation for the various restricted classes of SDSs considered in this paper. We also use this notation for restricted classes of SyDSs.

Table 1
Notation for restricted classes of SDSs

Notation	Interpretation
(BOOL, SYM)-SDS	Domain of state values is Boolean and each local transition function is symmetric.
(BOOL, MON)-SDS	Domain of state values is Boolean and each local transition function is monotone.
(BOOL, ST)-SDS	Domain of state values is Boolean and each local transition function is a simple-threshold function.
(BOOL, SIT)-SDS	Domain of state values is Boolean and each local transition function is either a simple-threshold function or a simple-inverted-threshold function.
(BOOL, AWT)-SDS	Domain of state values is Boolean and each local transition function is a (weighted) threshold function. The weights used in the local transition functions may be *asymmetric*; that is, the weights used by the local transition functions at the two nodes of an edge may not be equal.
(BOOL, SWT)-SDS	Domain of state values is Boolean, each local transition function is a (weighted) threshold function with *symmetric* weights; that is, for each edge, the weights used in the local transition functions at the two nodes of the edge are equal.

2.3. Problems considered

Throughout this paper, we assume that the domain of a given SDS (or SyDS) is finite and that each local transition function can be evaluated in polynomial time. The focus of this paper is on the following reachability problems for SDSs.

(1) Given an SDS $\mathscr{S}$ over a domain $\mathbb{D}$, two configurations $\mathscr{I}$, $\mathscr{B}$, and a positive integer t, the t-REACHABILITY problem is to decide whether $\mathscr{S}$ starting in configuration $\mathscr{I}$ will reach configuration $\mathscr{B}$ in t or fewer time steps. We assume that t is specified in binary. (If t is specified in unary, it is easy to solve this problem in polynomial time since we can execute $\mathscr{S}$ for t steps and check whether configuration B is reached at some step.)

(2) Given an SDS $\mathscr{S}$ over a domain $\mathbb{D}$ and two configurations $\mathscr{I}$, $\mathscr{B}$, the REACHABILITY problem is to decide whether $\mathscr{S}$ starting in configuration $\mathscr{I}$ ever reaches the configuration $\mathscr{B}$. (Note that, for $t \geqslant |\mathbb{D}|^n$, t-REACHABILITY is equivalent to REACHABILITY.)

(3) Given an SDS $\mathscr{S}$ over a domain $\mathbb{D}$ and a configuration $\mathscr{I}$, the FIXED POINT REACHABILITY problem is to decide whether $\mathscr{S}$ starting in configuration $\mathscr{I}$ reaches a fixed point.

In particular, we study how various restrictions on the local transition functions of an SDS affect the complexity of the above reachability problems. A summary of our results for these problems is given in Section 3.

3. Summary of results and their significance

As mentioned earlier, we study the complexity of several reachability problems for classes of SDSs over finite domains. Our results include lower bounds (i.e., *PSPACE*-hardness results) and upper bounds (i.e., polynomial time algorithms) on the complexity of reachability problems for SDSs under various restrictions on the local transition functions.

Lower bounds: Using a direct reduction from the acceptance problem for linear bounded automata (LBA), we show that the *t*-REACHABILITY, REACHABILITY and FIXED POINT REACHABILITY problems are *PSPACE*-complete for (BOOL, SIT)-SDSs. These results, in turn, allow us to show that the reachability problems remain *PSPACE*-complete for (BOOL, AWT)-SDSs as well. *PSPACE*-completeness results for (BOOL, MON)-SDSs follow as direct corollaries of our *PSPACE*-completeness results for (BOOL, AWT)-SDSs. Moreover, the results hold even under all of the following restrictions.

(a) The maximum node degree in the underlying graph is a constant.
(b) The pathwidth (and hence the treewidth) of the underlying graph is a constant.
(c) The number of distinct local transition functions used is a constant.

In addition to the above restrictions, for (BOOL, AWT)-SDSs, the hardness results hold even when the ratio of the maximum to the minimum weight is a constant. These hardness results can be viewed as indicating a trade-off between (i) asymmetry in information exchange between nodes (ii) the types of threshold functions that are sufficient to make the problems computationally intractable and (iii) the structure of the underlying graph.

Upper bounds: In contrast to the *PSPACE*-hardness results for (BOOL, AWT)-SDSs, we show that reachability problems for (BOOL, SWT)-SDSs can be solved in polynomial time when all the weights used in the local transition functions are positive and the ratio of the maximum weight to the minimum weight is bounded by a polynomial in the size of the SDS. Note that the class of (BOOL, ST)-SDSs is a subset of the class of (BOOL, SWT)-SDSs in which each weight is 1. Therefore, the result for (BOOL, SWT)-SDSs also allows us to conclude that the reachability problems for (BOOL, ST)-SDSs can be solved efficiently.

Applications: The results presented in this paper imply appropriate lower bounds for reachability problems under other dynamical system models. For instance, as observed in Section 7, the simplicity of the local transition functions and the sequential update mechanism can be used to immediately imply the *PSPACE*-hardness of appropriate reachability problems for *simple* classes of communicating finite state machines.

4. Related work

The SDS model for dynamical systems is related to two other models for dynamical systems, namely discrete Hopfield networks and cellular automata (CA). We discuss the relationship to discrete Hopfield networks first. In general, a discrete Hopfield

network consists of a directed graph with a state value from the domain $\{+1,-1\}$ for each node, a threshold for each node and a weight for each edge. The weights may not be symmetric. The next state of a node v is determined by a function of its current state, its threshold, the states of the neighbors which have an edge from v and the weights of those edges. Reachability problems for discrete Hopfield networks are known to be *PSPACE*-complete under the parallel state update model [10]. To the best of our knowledge, researchers have not considered how restrictions on the structure of the underlying graph affect the complexity of reachability problems for Hopfield networks. Our hardness results for SDSs hold even when the maximum node degree of the underlying graph is a constant. Using an elegant potential function argument, it has been shown [9,10] that when the edge weights and node thresholds are integers, Hopfield networks under sequential updates reach a fixed point regardless of the initial configuration. Our results show that SDSs whose local transition functions are threshold functions reach a fixed point when the threshold functions use the weights in a symmetric manner.

Computational aspects of CA have been studied by a number of researchers (see [25,37,16,15,35] and the references therein). However, most of the work addresses computability issues for infinite CA. Papers that are most relevant to our work are the following.

(1) Sutner [35] characterizes the complexity of reachability and predecessor existence problems for finite CA.
(2) Moore [25] makes an important connection between unpredictability of dynamical systems and undecidability of some of their properties. He argues that undecidability is a much stronger form of unpredictability.
(3) The work of Buss et al. [7] considers reachability problems for coupled automata. These automata do not interact with each other; instead, the system uses a global control rule which is independent of the identities of the automata. This identity independence assumption is similar to the use of symmetric Boolean functions in SDSs.

For additional references on discrete dynamical systems and Hopfield networks, see [2,8,16,29,37] and the references therein.

5. Reachability problems with threshold and monotone functions

5.1. Outline of results

This section establishes the complexity of reachability problems for several classes of SDSs over the Boolean domain. It can be verified that each of the reachability problems is in *PSPACE*. Therefore, the proofs address only the *PSPACE*-hardness aspect.

We first show (Section 5.2) that the reachability problems for (Bool, SIT)-SDSs are *PSPACE*-complete even when the maximum node degree of the underlying graph is a constant. Next, we show (Section 5.3) that the problems remain *PSPACE*-complete for (Bool, AWT)-SDSs which allow the local transition functions to use weights in an

asymmetric fashion. The *PSPACE*-completeness results for (BOOL, MON)-SDSs follow as direct corollaries of the results for (BOOL, AWT)-SDSs.

5.2. Hardness result for (BOOL, SIT)-SDSs

Theorem 5.1. *There exist constants d_2, p_2 and n_2 such that the t-*REACHABILITY, REACHABILITY *and* FIXED POINT REACHABILITY *problems are PSPACE-complete for* (BOOL, SIT)*-SDSs, even when all of the following restrictions hold*:
(a) *The maximum node degree in the underlying graph is bounded by d_2.*
(b) *The pathwidth* (*and hence the treewidth*) *of the underlying graph is bounded by p_2.*
(c) *The number of distinct local transition functions used is bounded by n_2.*

Proof. We prove the theorem by showing that (BOOL, SIT)-SDSs can simulate linear bounded automata (LBAs). Since the acceptance problem for LBAs is known to be *PSPACE*-complete [12], the theorem follows.

Our reduction consists of two steps. The first step simulates an LBA with a sequential circuit consisting of threshold gates and flip-flops, and the second step simulates the sequential circuit with a (BOOL, SIT)-SDS.

First, we recall that an LBA can be simulated by a linear array of finite automata (identical except for start state), where the linear array has a finite automaton for each cell of the LBA, and the next state of each cell is based on its current state and that of its immediate neighbors. Furthermore, each of the finite automata (and thus each cell of the LBA) can be implemented as a cyclic circuit as follows. In a sequential circuit, the values of state variables at one time period are specified as Boolean functions of the variables at the previous time. Our sequential circuit will use AND gates and OR gates to compute the new values of the state variables, and will use D-FLIP-FLOPs to store the values of the state variables. (At each time step, a D-FLIP-FLOP stores the Boolean value of its input.) We use an implementation where there is a pair of D-FLIP-FLOPs for each state variable: one D-FLIP-FLOP holding the value of the state variable, and the other holding the complement of the value. Thus, both complemented and uncomplemented literals are available to the combinational logic circuit that sets the D-FLIP-FLOPs. We assume that the equation for computing the new value for each D-FLIP-FLOP is given in a form that has only ANDs, ORs, and literals (uncomplemented and complemented state variables). By having the values of all the literals available, we have no need for NOT gates. Thus, the sequential circuit can be constructed using only AND, OR and D-FLIP-FLOPs. We also require that the input to each D-FLIP-FLOP is the output of some gate (even if the gate is a single-input OR gate). Thus, the circuit consists of a network of simple-threshold gates that set the new values of the D-FLIP-FLOP. It can be seen that the resulting sequential circuit simulates the given finite automaton and that the linear array of finite automata simulates the given LBA.

We now consider the simulation of the sequential circuit, denoted by S, by a (BOOL, SIT)-SDS, denoted by $\mathscr{S}_1$. First, we specify the underlying graph $G_1(V_1, E_1)$ of $\mathscr{S}_1$ beginning with V_1. We classify the nodes in V_1 as being either *mainstream*

nodes or *clock nodes*. We further partition the mainstream nodes into *computational nodes*, *driver nodes*, and *output nodes*. The driver nodes and the output nodes will be used to represent D-FLIP-FLOPS while the computational nodes will be used to represent the AND and OR circuit elements.

- For each gate in the combinational network for S, there is exactly one computational node (thus also a mainstream node) in V_1.
- For each D-FLIP-FLOP in S, there are exactly two mainstream nodes in V_1: a driver node and an output node. The driver nodes serve as inputs to the AND and OR gates of the sequential circuit. The output node is used to store the output of the D-FLIP-FLOP and thus all the output nodes together can be thought of as storing the configuration of the LBA at a given instant in time.
- Each mainstream node of V_1 has an associated set of clock nodes; each such set is referred to as the *clock* for that mainstream node.

The clock for each mainstream node contains a certain number of clock nodes which we refer to as the *impedance* of the mainstream node. The number of clock nodes depends on the type of mainstream node. For a computational node, the impedance is two plus the sum of the fan-in and fan-out of the corresponding gate in S. The impedance of a driver node is three plus the fan-out of the corresponding D-FLIP-FLOP. The impedance of an output node is 4. We now specify the edge set E_1 of G_1.

- For each connection in S from the output of a D-FLIP-FLOP to the input of a gate, there is an edge in E_1 between the driver node for that D-FLIP-FLOP and the computational node for that gate.
- For each pair of gates that are connected in circuit S, there is an edge between the corresponding computational nodes in G_1.
- For each connection in S from the output of a gate to the input of a D-FLIP-FLOP, there is an edge in E_1 between the computational node for that gate and the output node for that D-FLIP-FLOP.
- For each D-FLIP-FLOP, there is an edge between the driver node and the output node.
- Within each clock, there is an edge between each of the clock nodes and the mainstream node with which the clock is associated. Note that each clock node has an edge only to the mainstream node with which the clock is associated, so that the clock can be considered as "private" clock for its mainstream node. Also, note that the impedance of each mainstream node, which equals the number of its clock nodes, also equals two plus the number of other mainstream nodes connected to that mainstream node.

We now specify the set of local transition functions $\mathscr{F}_1$ of $\mathscr{S}_1$.

- The local transition functions for mainstream nodes are simple-threshold functions given as follows:
  - A computational node corresponding to a gate of S with threshold t has a simple-threshold function with threshold equal to t plus the number of nodes in its clock.
  - Each driver node and each output node have a simple-threshold function with threshold equal to one plus the number of nodes in its clock.
- For each clock node, the local transition function is the 1-simple-inverted-threshold function (i.e., the NOR function of its inputs).

Finally, we specify the permutation π_1 of $\mathscr{S}_1$. The nodes of $\mathscr{S}_1$ appear in the following order in π_1.

(1) The permutation begins with all the computational nodes, in an order that is a topological sort of the dataflow order in the combinational network of circuit S. (Thus, if the output of a gate x is an input to gate y, the computational node corresponding to x precedes the computational node corresponding to y.)
(2) All the driver nodes come next (in an arbitrary order, since there are no edges between driver nodes).
(3) All the output nodes are listed next (in an arbitrary order).
(4) All the clock nodes are listed next (in an arbitrary order).

This completes the construction of $\mathscr{S}_1$ from S. We now explain why the simulation works. Our simulation uses two transitions of $\mathscr{S}_1$ (i.e., two successive applications of the global transition function $F_{\mathscr{S}_1}$ of $\mathscr{S}_1$) to simulate one step of S.

We call a configuration of $\mathscr{S}_1$ *proper* if for each clock, all the nodes within that clock have the same value. If this value is 1, we say that the clock is ON, and if this value is 0, we say that the clock is OFF. In the simulation of S by $\mathscr{S}_1$, the configuration of $\mathscr{S}_1$ at the end of each step will be proper.

Consider a mainstream node x of $\mathscr{S}_1$ and its clock, during a step of $\mathscr{S}_1$. In accordance with permutation π_1, first the local transition function for x is evaluated, then later in the permutation, the local transition functions for the clock nodes associated with x are evaluated. Suppose that at the start of the step, x's clock is OFF. When the local transition function for x is evaluated, all its clock nodes have value 0. Since the threshold of the local transition function f_x at x is greater than the number of remaining inputs to that function (the inputs corresponding to x itself and to its neighbors that are also mainstream nodes), f_x evaluates to 0, regardless of the values of these remaining input variables. Next, for each node y within the clock, consider the evaluation of the corresponding local transition function f_y, which is the NOR function. Since node y and its neighbor x both have value 0, function f_y evaluates to 1. Thus, at the end of one transition of $\mathscr{S}_1$, mainstream node x has value 0, and its associated clock nodes have all been set to 1; that is, clock for x is now ON. Thus, the clock for each mainstream node is ON.

Now, suppose that at the start of a step of $\mathscr{S}_1$, a given clock is ON. As before, consider the clock's mainstream node x with its local transition function f_x. The clock nodes of x all have value 1, and so f_x will be sensitive to the values of the mainstream nodes that are neighbors of x. For each clock node y within x's clock, when the local transition function f_y (which is the NOR function) is evaluated, since y has value 1, f_y evaluates to 0. So, at the end of this step of $\mathscr{S}_1$, the mainstream node x has been set to a value that depends on the values of the mainstream neighbors of x at the time when f_x is evaluated, and the clock for x is now OFF.

We define a *configuration* of the circuit S to be an assignment of a Boolean value to each D-FLIP-FLOP in S. We now define the following mapping g from configurations of S into proper configurations of $\mathscr{S}_1$. For a configuration C of S, $g(C)$ is defined as follows.

- Each driver node of $\mathscr{S}_1$ has the value of the corresponding D-FLIP-FLOP of S.
- Each computational and output node of $\mathscr{S}_1$ has value 0.

- Each clock for a computational or output node of $\mathscr{S}_1$ is ON.
- Each clock for a driver node of $\mathscr{S}_1$ is OFF.

Suppose we use $F_{\mathscr{S}}$ to denote the global transition function of circuit S. Our key claim is the following.

Claim 1. *For every configuration C of S, $F_{\mathscr{S}_1}(F_{\mathscr{S}_1}(g(C)))=g(F_{\mathscr{S}}(C))$.*

Proof. Consider two steps of $\mathscr{S}_1$, beginning with configuration $g(C)$. In the first step of $\mathscr{S}_1$, the evaluation of the local transition functions of $\mathscr{S}_1$ results in the following:

(1) Each computational node x of $\mathscr{S}_1$ mimics the corresponding gate of S, and is set to the output value of that gate. (When the local transition function f_x of node x is evaluated, the neighboring mainstream nodes corresponding to the fan-in points of the gate corresponding to x in S have the same values as when the gate is evaluated in S, and the node itself and its neighbors corresponding to fan-out points have value 0. The threshold of f_x is the sum of the impedance and the threshold t of the corresponding gate of S. The number of clock nodes of x equals the impedance of x, and since x's clock is ON, these clock nodes all have value 1. Thus, f_x evaluates to 1 iff at least t of the mainstream nodes corresponding to the gate's fan-out points have value 1.)
(2) All the driver nodes are set to 0. (The clocks for the driver nodes are all OFF.)
(3) Each output node y is set to the new value of the corresponding D-FLIP-FLOP. (When the local transition function f_y of node y is evaluated, the clock is ON, the neighboring computational node corresponding to the gate whose output is the input to the D-FLIP-FLOP has the correct value, and the node itself and its neighboring driver node have value 0.)
(4) The clocks for computational nodes and output nodes are set to OFF, and the clocks for driver nodes are set to ON.

In the next step of $\mathscr{S}_1$, the evaluation of the local transition functions of $\mathscr{S}_1$ results in the following:

(1) All the computational nodes are set to 0. (Their clocks are all OFF.)
(2) Each driver node is set to the value of its neighboring output node. (When its local transition function is evaluated, the clock is ON, the neighboring output node has the new value of the corresponding D-FLIP-FLOP, and the node itself and its neighboring computational nodes all have value 0.)
(3) All the output nodes are set to 0. (Their clocks are all OFF.)
(4) The clocks for computational and output nodes are set to ON, and the clocks for driver nodes are set to OFF.

Thus, at the end of the second step of $\mathscr{S}_1$, each driver node has the value of the next state of the corresponding D-FLIP-FLOP of S, each computational and output node has value 0, each clock for a computational or output node is ON, and each clock for a driver node is OFF. This completes the proof of Claim 1. □

By induction on the number of steps in the operation of S, it can be seen from Claim 1 that if S starting in configuration I reaches configuration B in t steps, then $\mathscr{S}_1$ starting in configuration $g(I)$ reaches configuration $g(B)$ in $2t$ steps. Also note that

after an odd number of steps of $\mathscr{S}_1$, the configuration of $\mathscr{S}_1$ does not lie in the range of the mapping g. This completes the $PSPACE$-hardness proofs for the t-REACHABILITY and REACHABILITY problems for (BOOL, AWT)-SDSs.

By carrying out the above reduction starting from an LBA which when it accepts, reaches a certain state where it cycles forever, it can be seen that the REACHABILITY and FIXED POINT REACHABILITY problems for (BOOL, AWT)-SDSs are also $PSPACE$-hard.

Graph restrictions and local functions: We complete the proof of Theorem 5.1 by showing that the indicated restrictions on degree, pathwidth and local functions hold.

First, consider Step 1. Note that for any given LBA, the size of the constructed sequential circuit is linear in the number of cells of the LBA. Also, since the sequential circuit implements a linear array of finite automata, if the circuit is regarded as a graph with a node for each circuit element, the maximum node degree and the pathwidth (and hence the treewidth) of the graph are each bounded by a constant independent of the number of cells of the LBA.

Now consider Step 2. Our transformation replaces each AND, OR and D-FLIP-FLOP locally by a graph, whose size is bounded by a constant independent of the number of circuit elements. Moreover, the edge between two nodes corresponding to different circuit elements exists only if the circuit elements they replaced had an edge. Thus, the degree of each node and the pathwidth (and hence the treewidth) of the graph are bounded by constants.

Finally, consider the number of distinct local transition functions used. We use a simple-inverted threshold function at each clock node. Without loss of generality, we may assume that the fan-in and fan-out of each circuit element is a small constant (no more than 2). The type of threshold function at a mainstream node depends on the number of clock nodes it is adjacent to and the sum of fan-in and fan-out of the circuit element. Given that the total number of clock nodes per circuit element is bounded by a constant, we get that the number of threshold functions is bounded.

This completes the proof of Theorem 5.1. □

5.3. Hardness result for (BOOL, AWT)-SDSs

Recall that in (BOOL, AWT)-SDSs, each local transition function is a (weighted) threshold function and that the weights used by the local transition functions may be asymmetric. In this section, we show that the reachability problems remain $PSPACE$-complete for (BOOL, AWT)-SDSs. Our proof, which is based on a reduction from the reachability problems for (BOOL, SIT)-SDSs, essentially shows how simple-inverted-threshold functions can be simulated using threshold functions that use the weights in an asymmetric manner. In contrast, we will show in Section 6 that subject to some technical conditions, the reachability problems can be solved in polynomial time if asymmetry is not permitted.

Theorem 5.2. *There is a polynomial time reduction from a* (BOOL, SIT)*-SDS* $\mathscr{S} = (G, \mathscr{F})$ *and configurations* $\mathscr{I}$ *and* $\mathscr{B}$ *for* $\mathscr{S}$ *to a* (BOOL, AWT)*-SDS* $\mathscr{S}_1 = (G_1, \mathscr{F}_1)$ *and configurations* $\mathscr{I}_1$ *and* $\mathscr{B}_1$ *for* $\mathscr{S}_1$ *such that*

(1) *$\mathscr{S}$ starting in configuration $\mathscr{I}$ reaches $\mathscr{B}$ iff $\mathscr{S}_1$ starting in configuration $\mathscr{I}_1$ reaches $\mathscr{B}_1$. Moreover, for each t, $\mathscr{S}$ reaches $\mathscr{B}$ in t steps iff $\mathscr{S}_1$ reaches $\mathscr{B}_1$ in $t+1$ steps.*
(2) *$\mathscr{S}$ starting in configuration $\mathscr{I}$ reaches a fixed point iff $\mathscr{S}$ starting in $\mathscr{I}_1$ reaches a fixed point.*

Proof. Given the graph G, the graph G_1 of $\mathscr{S}_1$ is constructed as follows. Let $\{x_1, x_2, \ldots, x_n\}$ denote the nodes of G where the permutation $\pi = \langle x_1, x_2, \ldots, x_n\rangle$.

If the local transition function f_i associated with node x_i of G is a simple-threshold function, then corresponding to x_i, the graph G_1 will have two nodes, denoted by x_i and $\widehat{x_i}$. If the local transition function f_i associated with node x_i of G is a simple-inverted-threshold function, then corresponding to x_i, the graph G_1 will have three nodes, denoted by x_i, $\widehat{x_i}$ and x_i^{copy}.

The edges of G_1 are constructed as follows. Consider each edge $\{x_i, x_j\}$ of G. Corresponding to this edge, there are either two or four edges in G_1 as follows:
(a) If at least one of the local transition functions f_i and f_j is a simple-threshold function, then the two edges $\{x_i, x_j\}$ and $\{\widehat{x_i}, \widehat{x_j}\}$ are included in G_1.
(b) If at least one of the local transition functions f_i and f_j is a simple-inverted-threshold function, then the two edges $\{x_i, \widehat{x_j}\}$ and $\{\widehat{x_i}, x_j\}$ are included in G_1.

In addition to the edges specified above, for each i, $1 \leqslant i \leqslant n$, if the local transition function f_i associated with x_i in $\mathscr{S}$ is a simple-inverted-threshold function, then the three edges $\{x_i, \widehat{x_i}\}$, $\{\widehat{x_i}, x_i^{\text{copy}}\}$, and $\{x_i, x_i^{\text{copy}}\}$ are included in G_1.

The permutation π_1 for $\mathscr{S}_1$ is constructed as follows. Consider each node x_i of G. If f_i is a simple-threshold function, then let $\pi_i' = \langle x_i, \widehat{x_i}\rangle$. If f_i is a simple-inverted-threshold function, then let $\pi_i' = \langle x_i^{\text{copy}}, x_i, \widehat{x_i}\rangle$. Recall that the permutation π for $\mathscr{S}$ is $\langle x_1, x_2, \ldots, x_n\rangle$. Permutation π_1 for $\mathscr{S}_1$ is given by $\langle \pi_1', \pi_2', \ldots, \pi_n'\rangle$.

The local transition functions for the nodes of $\mathscr{S}_1$ are chosen as follows. Consider each node x_i of G. Let $N(x_i) = \{x_i, y_1, y_2, \ldots, y_r\}$ denote the neighbors of x_i in G, including x_i itself. Note that $|N(x_i)| = r+1$. Let f_i denote the local transition function associated with x_i in $\mathscr{S}$.

(a) Suppose f_i is the k_i-simple-threshold function. (That is, f_i is 1 iff at least k_i of its $r+1$ inputs are equal to 1.)
- The local transition function g_i associated with x_i in $\mathscr{S}_1$ is defined to be 1 if and only if at least k_i of the inputs from nodes x_i, $y_1, y_2, \ldots, y_r$ are equal to 1. (This is equivalent to saying that in the threshold function g_i, each of the $r+1$ inputs x_i, $y_1, y_2, \ldots, y_r$ has a weight of 1 and each of the other inputs has a weight of 0.)
- The local transition function $\widehat{g_i}$ associated with $\widehat{x_i}$ in $\mathscr{S}_1$ is defined to be 1 if and only if at least $r - k_i + 2$ of the inputs from nodes $\widehat{x_i}, \widehat{y_1}, \widehat{y_2}, \ldots, \widehat{y_r}$ are equal to 1.

(b) Suppose f_i is the k_i-simple-inverted-threshold function. (That is, f_i is 0 iff at least k_i of its $r+1$ inputs are equal to 1.)
- The local transition function g_i associated with x_i in $\mathscr{S}_1$ is defined to be 1 if and only if at least $r - k_i + 2$ of the inputs from nodes $\widehat{x_i}, \widehat{y_1}, \widehat{y_2}, \ldots, \widehat{y_r}$ are equal to 1.

- The local transition function $\widehat{g_i}$ associated with $\widehat{x_i}$ in $\mathscr{S}_1$ is defined to be 1 if and only if at least k_i of the inputs from nodes $x_i^{\text{copy}}, y_1, y_2, \ldots, y_r$ are equal to 1.
- The local transition function g_i^{copy} associated with x_i^{copy} in $\mathscr{S}_1$ is defined to be 1 if and only if the input from node x_i is equal to 1.

The intuition behind the above construction is as follows. When $\mathscr{S}_1$ executes, at each time step, each node x_i of $\mathscr{S}_1$ will have the same state value as the corresponding node x_i of $\mathscr{S}$. Further, the node $\widehat{x_i}$ of $\mathscr{S}_1$ will have the complement of the state value of x_i. The complement value is used to implement the simple-inverted-threshold functions of $\mathscr{S}$ using threshold functions in $\mathscr{S}_1$. Recall that when the local transition function f_i associated with x_i in $\mathscr{S}$ is a simple-inverted-threshold function, $\mathscr{S}_1$ also contains the node x_i^{copy}. The local transition function g_i^{copy} associated with node x_i^{copy} saves the (old) state value of x_i into x_i^{copy} before the state of x_i is updated. The threshold function associated with $\widehat{x_i}$ uses this saved value to ensure that in $\mathscr{S}_1$, at each time step, the state value of $\widehat{x_i}$ is the complement of the state value of x_i.

We need to specify the initial configuration $\mathscr{I}_1$ and the final configuration $\mathscr{B}_1$ for $\mathscr{S}_1$. To do this, we first define a function ψ that maps pairs of configurations $(\mathscr{X}, \mathscr{Y})$ of $\mathscr{S}$ into a configuration of $\mathscr{S}_1$. We specify configuration $\psi(\mathscr{X}, \mathscr{Y})$ of $\mathscr{S}_1$ by specifying the value each node of $\mathscr{S}_1$ has in that configuration, as follows. For $1 \leqslant i \leqslant n$, $\psi(\mathscr{X}, \mathscr{Y})(x_i) = \mathscr{X}(x_i)$, $\psi(\mathscr{X}, \mathscr{Y})(\widehat{x_i}) = \overline{\mathscr{X}(x_i)}$, and if $\mathscr{S}_1$ has the node x_i^{copy}, then $\psi(\mathscr{X}, \mathscr{Y})(x_i^{\text{copy}}) = \mathscr{Y}(x_i)$. (Here, we use $\overline{\mathscr{X}(x_i)}$ to mean the complement of the Boolean value $\mathscr{X}(x_i)$.)

Let us call a configuration $\mathscr{X}$ of $\mathscr{S}_1$ *proper* if for $1 \leqslant i \leqslant n$, $\mathscr{X}(x_i) = \overline{\mathscr{X}(\widehat{x_i})}$. The following is an easy observation.

Observation 5.1. *For any pair of configurations $(\mathscr{X}, \mathscr{Y})$ of $\mathscr{S}$, the configuration $\psi(\mathscr{X}, \mathscr{Y})$ of $\mathscr{S}_1$ is proper.* □

The next claim points out that when the starting configuration of $\mathscr{S}_1$ is proper, every subsequent configuration reached by $\mathscr{S}_1$ is also proper.

Claim 2. *Suppose $\mathscr{C}$ is a proper configuration of $\mathscr{S}_1$. Then $F_{\mathscr{S}_1}(\mathscr{C})$ is also a proper configuration.*

Proof. Recall that corresponding to each node x_i of $\mathscr{S}$, there is a group, say G_i, consisting of either two nodes $\{x_i, \widehat{x_i}\}$ or three nodes $\{x_i^{\text{copy}}, x_i, \widehat{x_i}\}$ in $\mathscr{S}_1$. For $1 \leqslant i \leqslant n$, let the ith substep of $\mathscr{S}_1$ consist of the state updates for the group G_i of nodes. We will argue that at the end of each substep, the configuration of $\mathscr{S}_1$ is proper. Since $\mathscr{S}_1$ starts in a proper configuration, the claim follows.

Assume that the configuration at the end of the $(i-1)$th substep is proper, and consider the ith substep. There are two cases to consider.

Case 1: $G_i = \{x_i, \widehat{x_i}\}$. In this case, the ith substep updates the two nodes of G_i in the order $\langle x_i, \widehat{x_i} \rangle$. Let $M(i) = \{x_i, y_1, y_2, \ldots, y_r\}$ denote the inputs to x_i which have weight 1 in the local transition function g_i. By our construction, the set $\hat{M}(i)$ of inputs to $\widehat{x_i}$ which have weight 1 in the local transition function $\hat{g}_i$ is given by $\hat{M}(i) = \{\widehat{x_i}, \widehat{y_1}, \widehat{y_2}, \ldots, \widehat{y_r}\}$.

Since the configuration of $\mathscr{S}_1$ at the end of substep $i-1$ is proper, the values of x_i and $\widehat{x_i}$ are complements of each other before the state of x_i is updated, and for $1 \leqslant j \leqslant r$, the values y_j and $\widehat{y_j}$ are complements of each other.

Therefore, for any $q \geqslant 0$, q of the values in $M(i)$ are 1 iff $r+1-q$ of the values in $\hat{M}(i)$ are 1. In particular, at least k_i of the values in $M(i)$ are 1 iff less than $r+2-k_i$ of the values in $\hat{M}(i)$ are 1. Recall that the thresholds of g_i and $\widehat{g_i}$ are k_i and $r+2-k_i$, respectively. Therefore, if the value of x_i at the end of substep i is 1 (0), then the value of $\widehat{x_i}$ at the end of substep i is 0 (1). In other words, the configuration at the end of the ith substep is proper.

Case 2: $G_i = \{x_i^{\text{copy}}, x_i, \widehat{x_i}\}$. In this case, the ith substep updates these nodes in the order $\langle x_i^{\text{copy}}, x_i, \widehat{x_i}\rangle$. By our construction, the set $M(i)$ of inputs with weight 1 to the local transition function g_i is given by $M(i) = \{\widehat{x_i}, \widehat{y_1}, \widehat{y_2}, \ldots, \widehat{y_r}\}$. Further, the set $\hat{M}(i)$ of inputs to $\widehat{x_i}$ which have weight 1 in the local transition function $\hat{g}_i$ is given by $\hat{M}(i) = \{x_i^{\text{copy}}, y_1, y_2, \ldots, y_r\}$. Further, the only input to the local transition function g_i^{copy} with weight 1 is x_i. Therefore, when the state of x_i^{copy} is updated, its new state value is the value of x_i at the end of the $(i-1)$th substep. This observation in conjunction with the assumption that the configuration of $\mathscr{S}_1$ before substep i is proper, implies that at the time when the local transition function $\widehat{g_i}$ is computed, the values of $\widehat{x_i}$ and x_i^{copy} are complements of each other, and for $1 \leqslant j \leqslant r$, the values y_j and $\widehat{y_j}$ are complements of each other. Therefore, for any $q \geqslant 0$, q of the values in $M(i)$ are 1 iff $r+1-q$ of the values in $\hat{M}(i)$ are 1. In particular, at least $r+2-k_i$ of the values in $M(i)$ are 1 iff less than k_i of the values in $\hat{M}(i)$ are 1. Recall that the thresholds of g_i and $\widehat{g_i}$ are $r-k_i+2$ and k_i, respectively. Therefore, if the value of x_i at the end of substep i is 1 (0), then the value of $\widehat{x_i}$ at the end of substep i is 0 (1). In other words, the configuration at the end of the ith substep is proper in this case also, and this completes the proof of Claim 2. □

Our next claim points out that $\mathscr{S}_1$ properly simulates $\mathscr{S}$.

Claim 3. *For any pair of configurations* $(\mathscr{X}, \mathscr{Y})$ *of* $\mathscr{S}$, $F_{\mathscr{S}_1}(\psi(\mathscr{X}, \mathscr{Y})) = \psi(F_{\mathscr{S}}(\mathscr{X}), \mathscr{X})$.

Proof. Let $F_{\mathscr{S}}(\mathscr{X}) = \mathscr{C}$, $F_{\mathscr{S}_1}(\psi(\mathscr{X}, \mathscr{Y})) = \mathscr{C}_1$, and $\psi(F_{\mathscr{S}}(\mathscr{X}), \mathscr{X}) = \mathscr{C}_2$.

(1) Consider any node x_i of $\mathscr{S}_1$, $1 \leqslant i \leqslant n$. We must show that $\mathscr{C}_1(x_i) = \mathscr{C}_2(x_i)$.

Recall that node x_i in $\mathscr{S}$ corresponds to node x_i of $\mathscr{S}_1$. There are two cases to consider.

Case 1: The local transition function f_i of x_i in $\mathscr{S}$ is a k_i-simple-threshold-function.

By our construction, the inputs with weight 1 to the local transition function g_i of x_i in $\mathscr{S}_1$ are exactly those inputs to the local transition function f_i in $\mathscr{S}$. Further, the mapping ψ ensures that each input with weight 1 to g_i has exactly the same value as that of the corresponding input to f_i, and the threshold of g_i is also k_i. Therefore, the values computed by f_i and g_i are identical. In other words, $\mathscr{C}_1(x_i) = \mathscr{C}(x_i)$. Now, by the definition of ψ, the value of $\mathscr{C}_2(x_i) = \psi(F_{\mathscr{S}}(\mathscr{X}), \mathscr{X})(x_i) = \mathscr{C}(x_i)$. Therefore, $\mathscr{C}_1(x_i) = \mathscr{C}_2(x_i)$ in this case.

Case 2: The local transition function f_i of x_i in $\mathscr{S}$ is a k_i-simple-inverted-threshold-function.

Let $M(i)=\{x_i, y_1, y_2, \ldots, y_r\}$ denote the inputs to the function f_i in $\mathscr{S}$. By our construction, the set $M_1(i)$ of inputs with weight 1 to the local transition function g_i of x_i in $\mathscr{S}_1$ is given by $M_1(i)=\{\widehat{x_i}, \widehat{y_1}, \widehat{y_2}, \ldots, \widehat{y_r}\}$. By Observation 5.1, $\psi(\mathscr{X}, \mathscr{Y})$ is a proper configuration of $\mathscr{S}_1$. So, each value in $M(i)$ is the complement of the corresponding value in $M_1(i)$. Therefore, for any $q \geqslant 0$, q values in $M(i)$ are 1 iff $r+1-q$ values in $M_1(i)$ are 1. In particular, less than k_i values in $M(i)$ are 1 iff at least $r+2-k_i$ values in $M_1(i)$ are 1. The threshold of g_i is $r+2-k_i$. Hence, the values computed by f_i and g_i are identical, and $\mathscr{C}_1(x_i)=\mathscr{C}_2(x_i)$ in this case also.

(2) Consider any node $\widehat{x_i}$ of $\mathscr{S}_1$, $1 \leqslant i \leqslant n$. We must show that $\mathscr{C}_1(\widehat{x_i})=\mathscr{C}_2(\widehat{x_i})$.

By Observation 5.1, $\psi(\mathscr{X}, \mathscr{Y})$ is a proper configuration of $\mathscr{S}_1$. Thus, from Claim 2, $\mathscr{C}_1=F_{\mathscr{S}_1}(\psi(\mathscr{X}, \mathscr{Y}))$ is also a proper configuration. That is, $\mathscr{C}_1(\widehat{x_i})=\overline{\mathscr{C}_1(x_i)}$. By the definition of the mapping ψ, $\mathscr{C}_2(\widehat{x_i})=\psi(F_{\mathscr{S}}(\mathscr{X}), \mathscr{X})(\widehat{x_i})=\overline{\mathscr{C}(x_i)}$. From the proof for (1) above, we know that $\mathscr{C}_1(x_i)=\mathscr{C}(x_i)$. Therefore, $\mathscr{C}_1(\widehat{x_i})=\mathscr{C}_2(\widehat{x_i})$.

(3) Consider any i, $1 \leqslant i \leqslant n$, for which $\mathscr{C}_1$ has the node x_i^{copy}. We must show that $\mathscr{C}_1(x_i^{\text{copy}})=\mathscr{C}_2(x_i^{\text{copy}})$.

As argued in the proof of Claim 2, x_i^{copy} saves the value of x_i before x_i is updated. Therefore, $\mathscr{C}_1(x_i^{\text{copy}})=\mathscr{X}(x_i)$. By the definition of the mapping ψ, $\mathscr{C}_2(x_i^{\text{copy}})=\psi(F_{\mathscr{S}}(\mathscr{X}), \mathscr{X})(x_i^{\text{copy}})=\mathscr{X}(x_i)$. Therefore, $\mathscr{C}_1(x_i^{\text{copy}})=\mathscr{C}_2(x_i^{\text{copy}})$.

This completes the proof of Claim 3. □

Recall that the initial and final configurations of $\mathscr{S}$ are $\mathscr{I}$ and $\mathscr{B}$, respectively. We can now specify the initial and final configurations $\mathscr{I}_1$ and $\mathscr{B}_1$ of $\mathscr{S}_1$ as follows. Let $\mathscr{I}^0$ denote the configuration of $\mathscr{S}$ where the state of every node of $\mathscr{S}$ is 0. We set $\mathscr{I}_1$ to be the configuration $\psi(\mathscr{I}, \mathscr{I}^0)$ and $\mathscr{B}_1$ to be $\psi(F_{\mathscr{S}}(\mathscr{B}), \mathscr{B})$. The following claim is an easy consequence of Claim 3 and induction on t.

Claim 4. *Let $\mathscr{S}$ be the given* (BOOL, SIT)*-SDS with initial and final configuration $\mathscr{I}$. Let $\mathscr{S}_1$ be the* (BOOL, AWT)*-SDS constructed as above with initial configuration $\mathscr{I}_1$. Then, for all $t \geqslant 1$, $\xi(\mathscr{S}_1, \mathscr{I}_1, t)=\psi(\xi(\mathscr{S}, \mathscr{I}, t), \xi(\mathscr{S}, \mathscr{I}, t-1))$.*

Theorem 5.2 is a direct consequence of Claim 4. □

We observe that the weights used in the local transition functions of the (BOOL, AWT)-SDS $\mathscr{S}_1$ constructed above are from $\{0,1\}$. By carrying out the reduction from a (BOOL, SIT)-SDS in which the number of distinct local transition functions is a constant and whose underlying graph has bounded degree and bounded pathwidth (treewidth), we obtain the following corollary:

Corollary 5.1. *There exist constants d_3, p_3 and n_3 such that the t-*REACHABILITY, REACHABILITY *and* FIXED POINT REACHABILITY *problems are PSPACE-complete for* (BOOL, AWT)*-SDSs, even when all of the following restrictions hold.*

(a) *The maximum node degree in the underlying graph is bounded by d_3.*
(b) *The pathwidth (and hence the treewidth) of the underlying graph is bounded by p_3.*
(c) *The number of distinct local transition functions used is bounded by n_3.*
(d) *The weights used in the local transition functions are from $\{0,1\}$.*

Further, we can replace the weight 0 in the proof of the above corollary by the value $1/(d_3+2)$, where d_3 is the constant denoting the maximum node degree in the underlying graph G_1 of SDS $\mathscr{S}_1$. With this weight, even if all the inputs which were previously assigned a weight of 0 were to be 1, the total contribution from all those inputs will still be less than 1; that is, the threshold function at any node will not be sensitive to any of the inputs with weight $1/(d_3+2)$. When each weight is from $\{1/(d_3+2), 1\}$, the ratio of the maximum to minimum weight is d_3+2, a constant. Thus, we also get the following corollary.

Corollary 5.2. *The reachability problems are* PSPACE-*complete for* (BOOL, AWT)-*SDSs which in addition to Conditions* (a), (b) *and* (c) *of Corollary* 5.1, *also satisfy the condition that the ratio of the largest to the smallest weight used in the local transition functions is a constant.*

The above corollary points out that asymmetric weights are sufficient to make the reachability problems for (BOOL, AWT)-SDSs *PSPACE*-complete; a large ratio of maximum to minimum weights is not needed.

It can be seen that each threshold function that uses only nonnegative weights is a monotone function. Therefore, we also get the following:

Corollary 5.3. *The reachability problems are* PSPACE-*complete for* (BOOL, MON)-*SDSs even when Conditions* (a), (b) *and* (c) *of Corollary* 5.1 *hold.*

6. Polynomial time algorithms for (BOOL, SWT)-SDSs

As mentioned in Section 2.2, in (BOOL, SWT)-SDSs, the local transition functions (which are threshold functions) use weights in a symmetric manner. More precisely, for each edge $\{x, y\}$ of the SDS, the weight of the input corresponding to y used in the local transition function f_x is equal to the weight of the input corresponding to x used in the local transition function f_y. Therefore, we envision this weight as being associated with the edge $\{x, y\}$, and denote it by $w(\{x, y\})$. Also, for each node x, the weight of the input corresponding to x used in f_x can be envisioned as the weight of the node x, denoted by $w(x)$.

In this section, we show that the reachability problems remain efficiently solvable for (BOOL, SWT)-SDSs, provided all the weights are strictly positive and the ratio of the largest weight to the smallest weight is bounded by a polynomial in the size of the given SDS.

Throughout Section 6, $\mathscr{S}$ is a (BOOL, SWT)-SDS in which all the node and edge weights are strictly positive. For a node v, let k_v denote the threshold value for the local transition function f_v, and let $W(v)$ denote the sum of the weight of v and the weights of all the edges incident on v. If $k_v < 0$, then since all the node and edge weights are strictly positive, f_v is the constant function which is 1 for all inputs. This function can be realized by setting $k_v = 0$. Further, if $k_v > W(v)$, then f_v is the constant function which is 0 for all inputs. This function can realized by setting $k_v = W(v) + \varepsilon$,

for any $\varepsilon > 0$. Therefore, we assume that for each node v, $0 \leqslant k_v \leqslant W(v) + \varepsilon$, where the value $\varepsilon > 0$ can be chosen appropriately.

For each node v, define $T_1(v) = k_v$ and $T_0(v) = W(v) - k_v + \varepsilon$. Since $0 \leqslant k_v \leqslant W(v) + \varepsilon$, both $T_1(v)$ and $T_0(v)$ are nonnegative. The value $T_1(v)$ can be envisioned as the minimum total weight of the 1-inputs to the local transition function f_v such that the output of f_v is 1. Similarly, the value $T_0(v)$ can be envisioned as the minimum total weight of the 0-inputs to the local transition function f_v such that the output of f_v is 0. Note that for any node v, $T_1(v)$ and $T_0(v)$ are nonnegative even if the function f_v is a constant function (i.e., f_v is 0 for all inputs or f_v is 1 for all inputs).

Theorem 6.1. *Let $\mathcal{S}$ be a* (BOOL, SWT)*-SDS in which all the node and edge weights are strictly positive. Let $w_{\max}$ denote the largest value among the node and edge weights, and let $w_{\min}$ denote the smallest node weight. If the ratio $w_{\max}/w_{\min}$ is bounded by a polynomial in the size of $\mathcal{S}$, then the* t*-*REACHABILITY, REACHABILITY *and* FIXED POINT REACHABILITY *problems for $\mathcal{S}$ can be solved in polynomial time.*

Proof. We use a potential function argument. Given a (BOOL, SWT)-SDS $\mathcal{S}$ and a configuration $\mathscr{C}$ for $\mathcal{S}$ we assign a potential to each node and each edge in the underlying graph G as discussed below.

The potential $P(\mathscr{C}, v)$ of a node v with respect to configuration $\mathscr{C}$ is defined as follows:

$$P(\mathscr{C}, v) = \begin{cases} T_1(v) & \text{if } \mathscr{C}(v) = 1, \\ T_0(v) & \text{if } \mathscr{C}(v) = 0. \end{cases}$$

The potential $P(\mathscr{C}, e)$ of an edge $e = \{u, v\}$ with respect to configuration $\mathscr{C}$ is defined as follows:

$$P(\mathscr{C}, e) = \begin{cases} w(e) & \text{if } e = \{u, v\} \text{ and } \mathscr{C}(u) \neq \mathscr{C}(v), \\ 0 & \text{otherwise.} \end{cases}$$

For a configuration $\mathscr{C}$, the potential of $\mathcal{S}$ is defined by

$$P(\mathscr{C}, \mathcal{S}) = \sum_{v \in V} P(\mathscr{C}, v) + \sum_{e \in E} P(\mathscr{C}, e).$$

The following claim gives upper and lower bounds on the potential of $\mathcal{S}$ for any configuration $\mathscr{C}$.

Claim 5. *For any configuration $\mathscr{C}$, $0 \leqslant P(\mathscr{C}, \mathcal{S}) \leqslant (3m + n)w_{\max} + n\varepsilon$, where n and m are, respectively, the number of nodes and edges of G.*

Proof. First, consider the lower bound. As observed earlier, for any node v, the values $T_0(v)$ and $T_1(v)$ are both nonnegative. Thus, $P(\mathscr{C}, v) \geqslant 0$ for all $v \in V$. Since the edge weights are strictly positive, $P(\mathscr{C}, e) \geqslant 0$ for all $e \in E$. Thus, $P(\mathscr{C}, \mathcal{S}) \geqslant 0$.

We prove the upper bound on the potential of $\mathcal{S}$ by considering the contributions of the nodes and edges separately. For any node v, $T_0(v)$ and $T_1(v)$ are each bounded

by $W(v)+\varepsilon$. Therefore, $P(\mathscr{C},v)\leqslant W(v)+\varepsilon$. For each $v\in V$, let δ_v denote the degree of v. Since the weight of each edge and node is at most $w_{\max}$, and $W(v)$ also includes the weight $w(v)$ of the node v, we have $W(v)\leqslant(\delta_v+1)w_{\max}$. Therefore,

$$\begin{aligned}\sum_{v\in V} P(\mathscr{C},v) &\leqslant \sum_{v\in V}[(\delta_v+1)w_{\max}+\varepsilon]\\ &= (2m+n)w_{\max}+n\varepsilon. \end{aligned} \tag{1}$$

For any edge e, $P(\mathscr{C},e)\leqslant w(e)\leqslant w_{\max}$. Therefore,

$$\sum_{e\in E} P(\mathscr{C},e) \leqslant mw_{\max}. \tag{2}$$

From Eqs. (1) and (2), it follows that $P(\mathscr{C},\mathscr{S})\leqslant(3m+n)w_{\max}+n\varepsilon$. This completes the proof of Claim 5. □

The next claim points out that the potential of $\mathscr{S}$ *decreases* whenever the state of a node changes. In stating the claim, we think of each step of $\mathscr{S}$ as consisting of n substeps, where each substep involves the evaluation of the local transition function at a node and updating the state of that node.

Claim 6. *Suppose v is a node of $\mathscr{S}$ that undergoes a state change during a step of $\mathscr{S}$. Let $\mathscr{C}$ and $\mathscr{C}'$ denote respectively the configuration of $\mathscr{S}$ just prior to the substep and just after the substep in which the state of v changes. Then, $P(\mathscr{C}',\mathscr{S})\leqslant P(\mathscr{C},\mathscr{S})-w_{\min}$.*

Proof. Let the state of node v change from a to b in the substep under consideration. As part of this substep, when f_v is evaluated, let $W_a(v)$ denote the total weight of all the edges between v and a neighbor of v whose state value is a. Similarly, let $W_b(v)$ denote the total weight of all the edges between v and a neighbor of v whose state value is b. When the state of v changes from a to b, only the potential of node v and the potentials of the edges incident on v may change; the potentials of other nodes and edges of G are unaffected.

Before the change in the state of v, let $\sigma(v)$ denote the sum of the potential of v and the potentials of the edges incident on v. Clearly, $\sigma(v)=T_a(v)+W_b(v)$. Let $\sigma'(v)$ denote the sum of the potential of v and the potentials of the edges incident on v after the change in the state of v. As before, $\sigma'(v)=T_b(v)+W_a(v)$. Since the state of v changed from a to b, we have $W_b(v)\geqslant T_b(v)$ and $W_a(v)+w(v)<T_a(v)$. Consequently, $\sigma(v)\geqslant T_a(v)+T_b(v)$ and $\sigma'(v)<T_a(v)+T_b(v)-w(v)$. Thus, the decrease in potential due to the change in state of v is $\sigma(v)-\sigma'(v)>w(v)\geqslant w_{\min}$. Claim 6 follows. □

We now continue with the proof of Theorem 6.1. From Claim 5, the initial potential of the SDS $\mathscr{S}$ is at most $P_{\max}=(3m+n)w_{\max}+n\varepsilon$. Each state change decreases the potential by at least $w_{\min}$. Since the total potential of the SDS is always nonnegative, after at most $\lfloor P_{\max}/w_{\min}\rfloor$ steps of the SDS, no state changes can occur; that is, the SDS reaches a fixed point.

Let $\beta=(3m+n)w_{\max}/w_{\min}$ and $\gamma=\beta-\lfloor\beta\rfloor$. Note that $\gamma<1$, and that

$$P_{\max}/w_{\min}=\lfloor(3m+n)w_{\max}/w_{\min}\rfloor+\gamma+n\varepsilon/w_{\min}.$$

By choosing ε so that $0<\varepsilon<(1-\gamma)w_{\min}/n$, we have $\gamma+n\varepsilon/w_{\min}<1$. This ensures that $\lfloor P_{\max}/w_{\min}\rfloor=\lfloor(3m+n)w_{\max}/w_{\min}\rfloor$, which in turn, is an upper bound the number of steps before $\mathscr{S}$ reaches a fixed point.

Since the ratio $w_{\max}/w_{\min}$ is bounded by a polynomial in the size of the SDS, $\mathscr{S}$ reaches a fixed point within a polynomial number of steps. Thus, the REACHABILITY, t-REACHABILITY and FIXED POINT REACHABILITY problems for the SDS $\mathscr{S}$ can be solved in polynomial time. □

7. Applications: Hopfield networks and communicating finite state machines

We discuss briefly how our results can be directly used to imply appropriate lower bounds for classes of Hopfield networks and communicating finite state machines.

Hopfield networks: As discussed earlier, our lower bounds for reachability problems for (BOOL, AWT)-SDSs directly imply that reachability problems for Hopfield networks with sequential update and asymmetric weights are *PSPACE*-hard. Moreover, the result holds for very small edge weights, bounded degree and bounded pathwidth (and hence treewidth) graphs. To our knowledge, such results have not been reported earlier. Our model of SDSs with simple-threshold and simple-inverted-threshold functions and the corresponding *PSPACE*-hard lower bounds for the reachability problems suggest a potentially new variant of Hopfield networks.

Communicating Finite State Machines (*CFSMs*): CFSMs have been widely studied as models of concurrent processes. As a result, a number of models have been proposed in the literature. Since these models were proposed for different applications, they are not always equivalent. We refer the reader to [1,6,14,17,18,24,30,28,34,36] for definitions, results, applications and the state of current research in this area. The basic setup consists of a collection of finite state machines. These machines communicate with each other via explicit channels [6,28,14] or via action symbols [30,34]. Our results apply to both these variants. To see this, we note the following:

(1) Simple-threshold and simple-inverted-threshold functions can be easily represented as finite state machines (FSMs) that essentially emulate a counter. The FSM corresponding to each node of an SDS consists of two parts, namely a control part and a part simulating the threshold function. (For some models, we can sometimes eliminate the control part.)

(2) Sequential update of the nodes of an SDS can be simulated by using n distinct (one for each machine) action symbols that in effect imply that each FSM is updated in the order determined from the ordering used for the given SDS. When dealing with explicit channels, this can be done by initializing all the FIFO I/O channels and using the control part to make sure that each machine corresponding to a threshold function makes a transition only after all its inputs have been received. At that point, the transition simply consists of counting how many inputs are 1

and how many are 0. After this, the machine posts the result of evaluating the function on each of its output channels.

Given these observations, the remaining details of the simulation are fairly straightforward. Our results show that the *PSPACE*-hardness results for reachability problems for CFSMs hold for extremely simple individual automata communicating using very simple rules. Thus, the results extend some of the earlier results in [34,30] on the complexity reachability problems for communicating state machines.

8. Summary and conclusions

We showed that the reachability problems for SDSs, where each local transition function is either a simple-threshold function or a simple-inverted-threshold function, are *PSPACE*-complete. We also showed that these intractability results extend to SDSs with monotone local transition functions and to other dynamical system models such as Hopfield networks and communicating finite state machines.

Additional results for other restricted SDSs are reported in [4]. For example, it is shown in [4] that for SDSs over a unitary semiring with linear local transition functions, the t-REACHABILITY problem can be solved in polynomial time. It is also shown that every Boolean SDS where each local transition function is the 3-simple-threshold function reaches a fixed point in at most $\lfloor 3n/2 \rfloor$ steps, where n is the number of nodes in the underlying graph.

Our results for SDSs with threshold functions provide one way of delineating classes of SDSs for which reachability problems are intractable and the classes of SDSs for which reachability problems are efficiently solvable. However, an exact characterization of the complexity of reachability problems for SDSs remains an intriguing open problem.

Acknowledgements

We thank the referees for providing a number of valuable suggestions. This research has also been funded in part by the LDRD-DR project *Foundations of Simulation Science* and by the LDRD-ER project *Extremal Optimization* at the Los Alamos National Laboratory. We thank Gabriel Istrate, Leonid Gurvits and Paul Wollan for drawing our attention to the results on Hopfield networks.

References

[1] R. Alur, S. Kannan, M. Yannakakis, Communicating hierarchical state machines, Proc. 26th Internat. Colloq. on Automata, Languages, and Programming (ICALP), Springer, Berlin, 1999, pp. 169–178.

[2] E. Asarin, O. Maler, On some relations between dynamical systems and transition systems, Proc. 21st Internat. Colloq. on Automata, Languages and Programming (ICALP), Jerusalem, Israel, Lecture Notes in Computer Science, Vol. 820, Springer, Berlin, July 1994, pp. 59–72.

[3] C. Barrett, H. Hunt III, M. Marathe, S. Ravi, D. Rosenkrantz, R. Stearns, Analysis problems for sequential dynamical systems and communicating state machines, in: Proc. Internat. Symp. on

Mathematical Foundations of Computer Science (MFCS'01), Marianske Lazne, Czech Republic, August 2001, Lecture Notes in Computer Science, Vol. 2136, Springer, Berlin, pp. 159–172.
[4] C. Barrett, H. Hunt III, M. Marathe, S. Ravi, D. Rosenkrantz, R. Stearns, Further analysis of sequential dynamical systems with restricted local transition functions, February 2002, in preparation.
[5] C. Barrett, H. Mortveit, C. Reidys, Elements of a theory of computer simulation III: equivalence of SDS, Appl. Math. Comput. 122 (2001) 325–340.
[6] D. Brand, P. Zafiropulo, On communicating finite-state machines, J. Assoc. Comput. Mach. 30 (2) (1983) 323–342.
[7] S. Buss, C. Papadimitriou, J. Tsitsiklis, On the predictability of coupled automata: an allegory about chaos Complex Systems 1 (5) (1991) 525–539.
[8] K. Cullik, S. Yu, Undecidability of CA classification schemes, Complex Systems 2 (2) (1988) 177–190.
[9] P. Floréen, E. Goles, G. Weisbuch, Transient length in sequential iterations of threshold functions, Discrete Appl. Math. 6 (1983) 95–98.
[10] P. Floreén, P. Orponen, Complexity issues in discrete Hopfield networks, in: I. Parberry (Ed.), Computational and Learning Complexity of Neural Networks: Advanced Topics, forthcoming.
[11] P. Gacs, Deterministic computations whose history is independent of the order of asynchronous updating, Technical Report, Computer Science Department, Boston University, 1997.
[12] M.R. Garey, D.S. Johnson, Computers and Intractability: A Guide to the Theory of NP-completeness, Freeman, San Francisco, CA, 1979.
[13] E. Goles, On dynamics of positive automata networks, Theoret. Comput. Sci. 41 (1985) 19–32.
[14] M. Gouda, C. Chang, Proving liveness for networks of communicating finite state machines, ACM Trans. Programming Languages Systems (TOPLAS) 8 (1) (1986) 154–182.
[15] F. Green, NP-complete problems in cellular automata, Complex Systems 1 (3) (1987) 453–474.
[16] H. Gutowitz (Ed.), Cellular Automata: Theory and Experiment, North-Holland, Amsterdam, 1989.
[17] D. Harel, O. Kupferman, M.Y. Vardi, On the complexity of verifying concurrent transition systems, Proc. 8th Internat. Conf. on Concurrency Theory (CONCUR'97), Warsaw, Poland, July 1997, Lecture Notes in Computer Science, Vol. 1243, 1997, pp. 258–272.
[18] C. Hoare, Communicating Sequential Processes, Prentice-Hall, Englewood Cliffs, NJ, 1984.
[19] J. Hopfield, Neural networks and physical systems with emergent collective computational abilities, Proc. Nat. Acad. Sci. U.S.A. 81 (1982) 3088–3092.
[20] B. Huberman, N. Glance, Evolutionary games and computer simulations, Proc. Nat. Acad. Sci. 90 (1993) 7716–7718.
[21] Z. Kohavi, Switching and Finite Automata Theory, McGraw-Hill Book Company, New York, 1970.
[22] R. Laubenbacher, B. Pareigis, Finite dynamical systems, Technical Report, Department of Mathematical Sciences, New Mexico State University, Las Cruces.
[23] B. Martin, A geometrical hierarchy of graph via cellular automata, in: Th. Worsch, R. Wolmar (Eds.), Proc. Mathematical Foundations of Computer Science (MFCS'98): Satellite Workshop on Graph Automata, Universität Karlsruhe, 1998.
[24] R. Milner, Communicating and Mobile systems: the π-calculus, Cambridge University Press, Cambridge, 1999.
[25] C. Moore, Unpredictability and undecidability in dynamical systems, Phys. Rev. Lett. 64 (20) (1990) 2354–2357.
[26] H. Mortveit, C. Reidys, Discrete sequential dynamical systems, Discrete Math. 226 (2001) 281–295.
[27] C. Nicthiu, E. Remila, Simulations of graph automaton, in: Th. Worsch, R. Wolmar (Eds.), Proc. Mathematical Foundations of Computer Science (MFCS'98), Satellite Workshop on Graph Automata, Universität Karlsruhe, 1998, pp. 69–78.
[28] W. Peng, Deadlock detection in communicating finite state machines by even reachability analysis, Mobile Networks (MONET) 2 (3) (1997) 251–257.
[29] G. Pighizzini, Asynchronous automata versus asynchronous cellular automata, Theoret. Comput. Sci. 132 (1–2) (1994) 179–207.
[30] A. Rabinovich, Complexity of equivalence problems for concurrent systems of finite agents, Inform. and Comput. 127 (2) (1997) 164–185.
[31] C. Reidys, Sequential dynamical systems: phase space properties, Adv. Appl. Math., to appear.

[32] Z. Roka, One-way cellular automata on Cayley graphs, Theoret. Comput. Sci. 132 (1–2) (1994) 259–290.
[33] C. Schittenkopf, G. Deco, W. Brauer, Finite automata-models for the investigation of dynamical systems, Inform. Process. Lett. 63 (3) (1997) 137–141.
[34] S.K. Shukla, H.B. Hunt III, D.J. Rosenkrantz, R.E. Stearns, On the complexity of relational problems for finite state processes, Internat. Colloq. on Automata Programming and Languages (ICALP), 1996, pp. 466–477.
[35] K. Sutner, On the computational complexity of finite cellular automata, J. Comput. System Sci. 50 (1) (1995) 87–97.
[36] M. Vardi, P. Wolper, An automata-theoretic approach to automatic program verification, Proc. 1st IEEE Symp. on Logic in Computer Science, 1986, pp. 332–344.
[37] S. Wolfram (Ed.), Theory and Applications of Cellular Automata, World Scientific, Singapore, 1987.